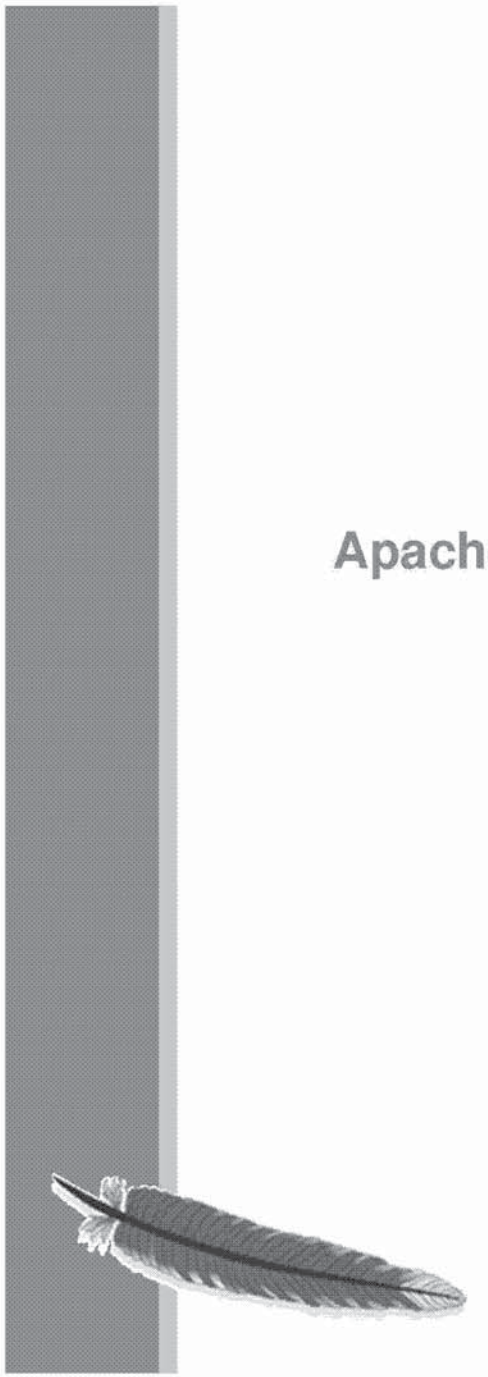
Exhibit 20



JavaOne

Apache Harmony: An Open Innovation

Tim Ellison
Apache Harmony Project

http://harmony.apache.org

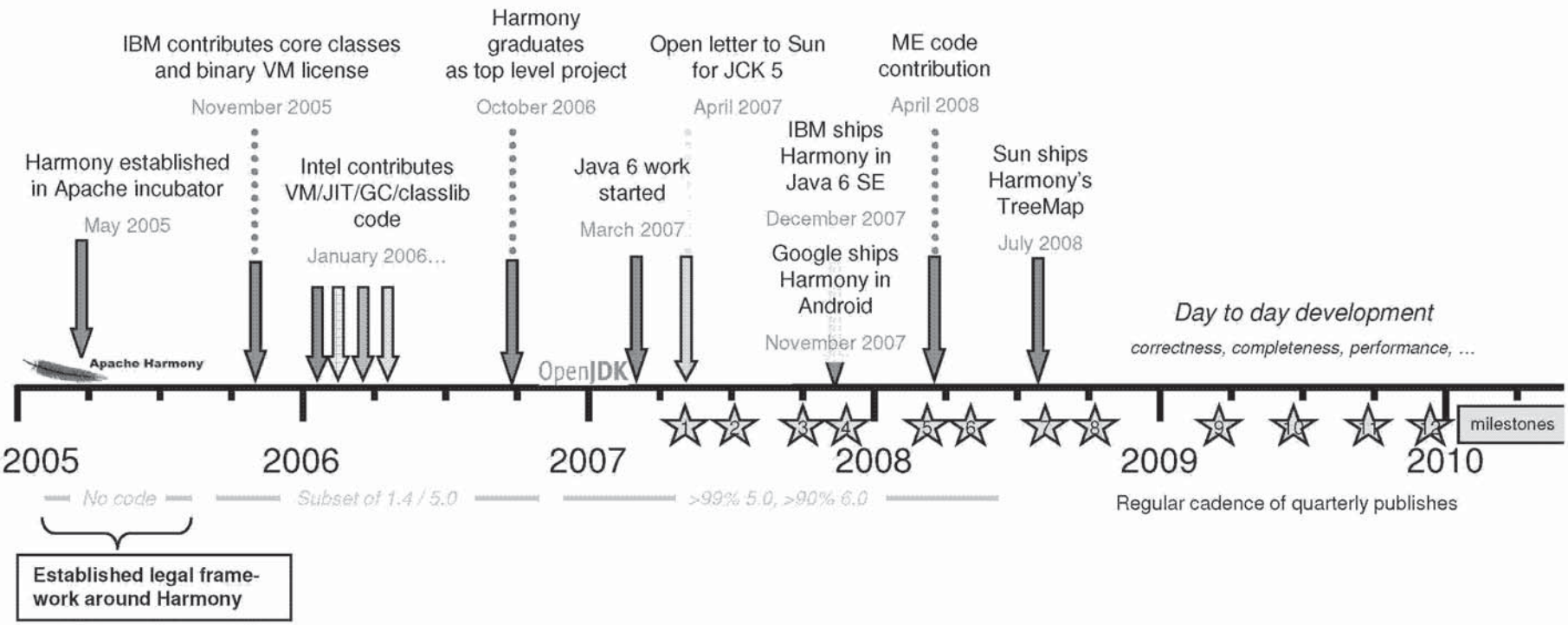
Join me during the next 45 mins to…

Learn about the motivations, current status, and future plans of the Apache Harmony project.

Agenda

- Project History
- Development Model
- Code and Runtime Modularity
- How Are We Doing?
- Demo
- Harmony and the JCK
- Wrap-up

Apache Harmony – a brief history



- **Primary goals**
  - Protect IP rights of the Java ecosystem.
  - Independent implementation of virtual machines, related class libraries, and other software commonly associated with Java platform implementations.
  - Community-developed, modular architecture allowing sharing and innovation.

Enhanced IP-cleanliness

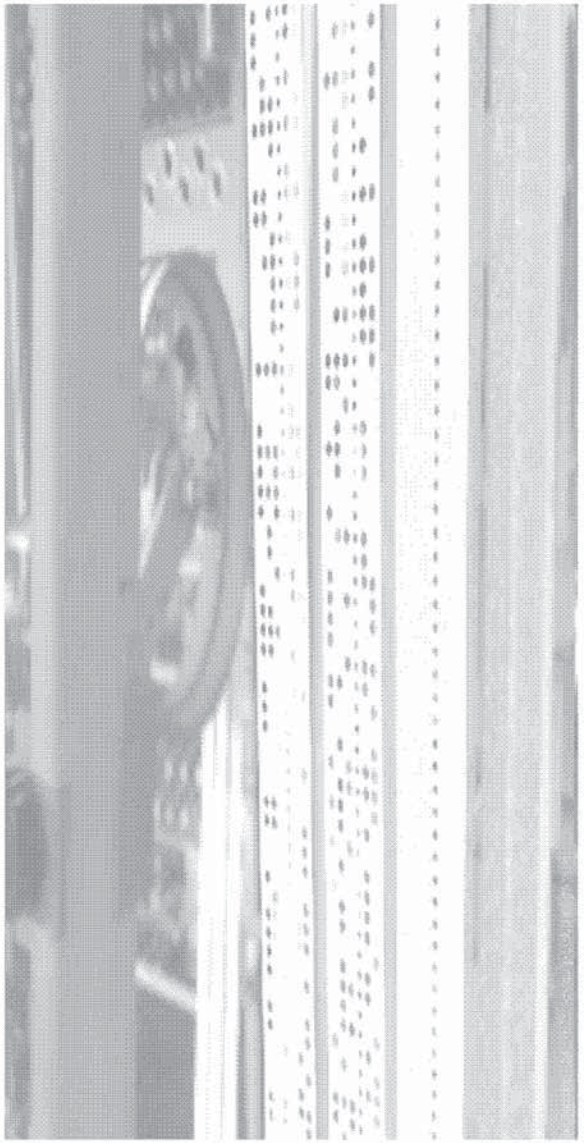
- Contributors detail their prior access via a project questionnaire.
- Developers can contribute in functional areas where they have not studied closed-source implementations (exceptions apply).
- Existing code being contributed to the project must provide acceptable pedigree information.

This is in *addition* to the standard Apache contribution processes

See: http://harmony.apache.org/auth_cont_quest.html

Bootstrapping the project

- Establish a common goal
  - compliant and compatible Java SE implementation
- JVM contributions
  - initial VM implementations, … from Cobbs and Lydick
  - DRLVM, … from Intel
- Class library code contributions
  - core lang, util, net, IO, beans, … from IBM
  - security, print, sound, awt, swing, … from Intel
  - math, rmi, … from National University of Cordoba, Argentina
- Production-quality VMs for build and test
  - IBM J9 and BEA JRockit were made available to project
    - Used for development
    - Not under open source license, not a contribution

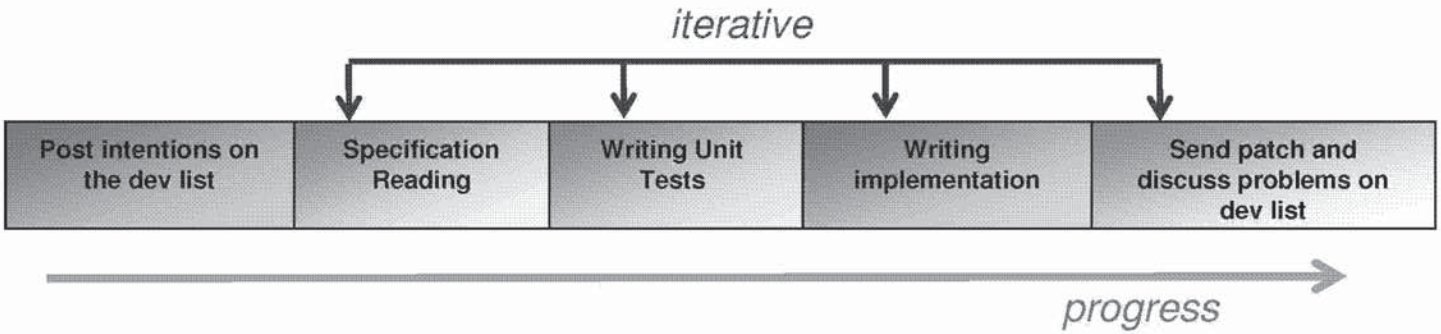


Agenda

- Project History
- Development Model
- Code and Runtime Modularity
- How Are We Doing?
- Demo
- Harmony and the JCK
- Wrap-up

Spec-driven development

Producing a compliant implementation

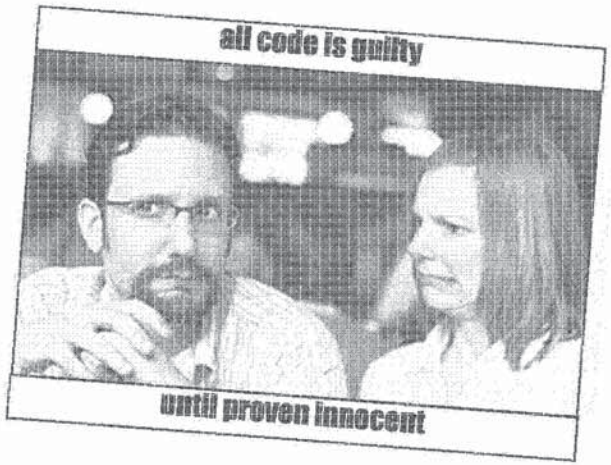


- Implementing the Java specifications as found in
  - Java SE platform Javadoc
  - Java Programming Language Specification
  - Java Virtual Machine (JVM) Specification, etc.
- Ambiguities and omissions...
  - resolved by consulting the Reference Implementation
  - determined by functional API testing

Test-driven development

Producing a robust, compatible implementation

- Functional testing
  - API tests, internal interfaces tests, bug regression tests
  - Component oriented
- Integration testing
  - Build verification, test with different VMs
  - Component assembly oriented
- Application / System testing
  - Running popular applications, *ad hoc* and test suites
  - End-product and compatibility oriented
- Platform, Performance and Stress testing
  - Multi-platform continuous integration, performance / stress suites
  - Robustness and quality oriented



Project repository organization

/classlib

- the class library code
- branches for Java 5 and Java 6

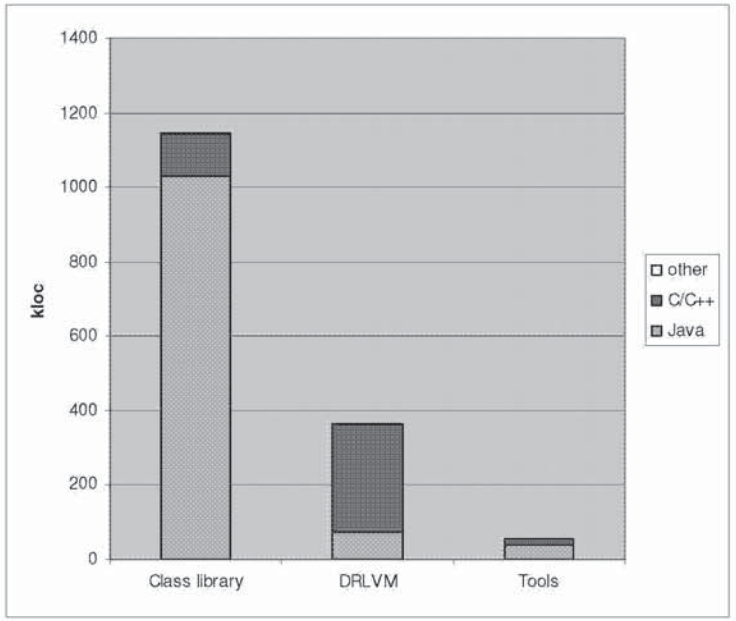
/drlvm

- the VM, JIT, GC

/jdktools

- Java technology development tools (javac, javah, javap, ...)

Java 5 code base is >1.5 Million non-comment lines of code



generated using David A. Wheeler's 'SLOCCount'

Federated workspace build

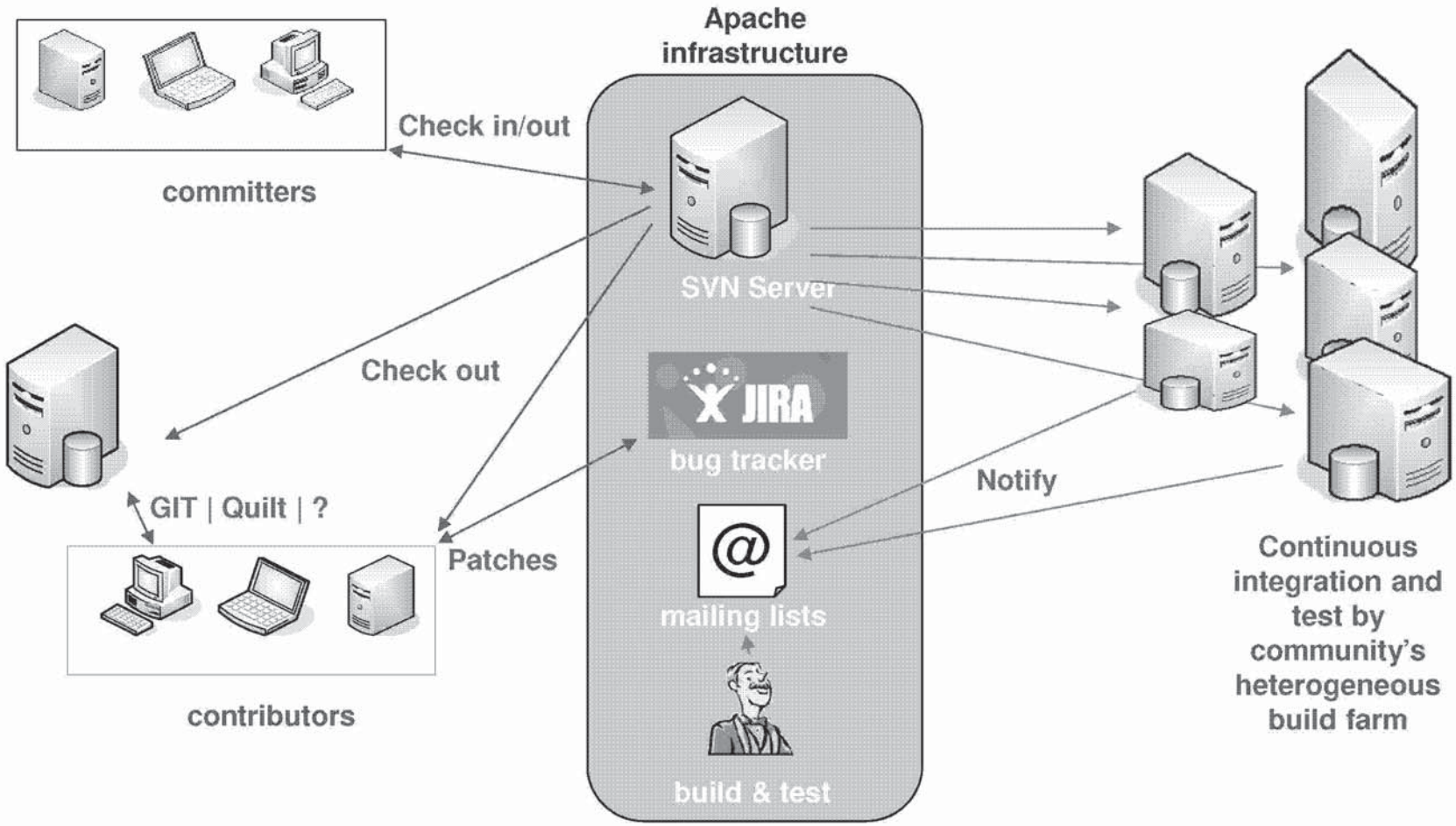
Bringing it all together

```
/trunk
   build.xml
   /classlib          -> class libraries
   /drlvm             -> virtual machine, JIT, GC
   /jdktools          -> developer tools
   /common_resources  -> shared resources
```

- Single command build, just type "`ant`"
- Downloads project dependencies
- Successful build results in:
  Harmony Development Kit (HDK) – for implementation developers,
  Java Development Kit (JDK) – for Java developers,
  Java Runtime Environment (JRE) – for Java users.

Distributed build-test

Heterogeneous build and test farm across community

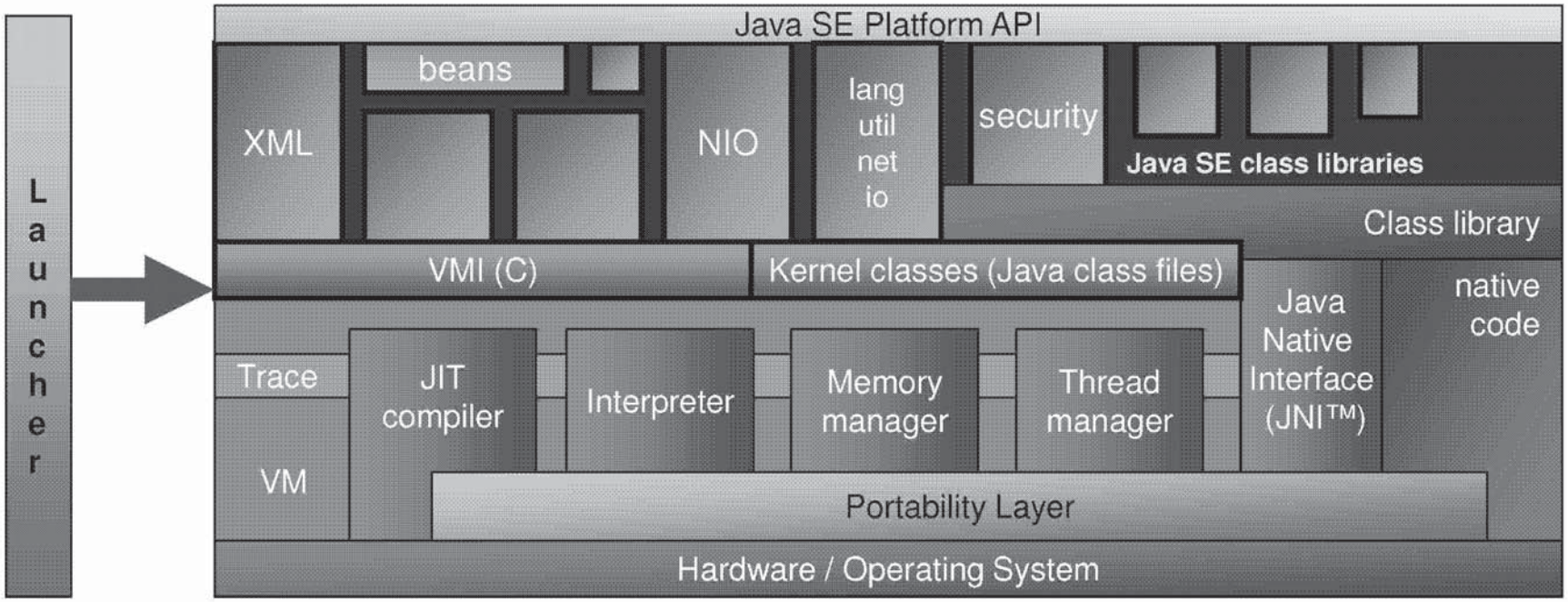


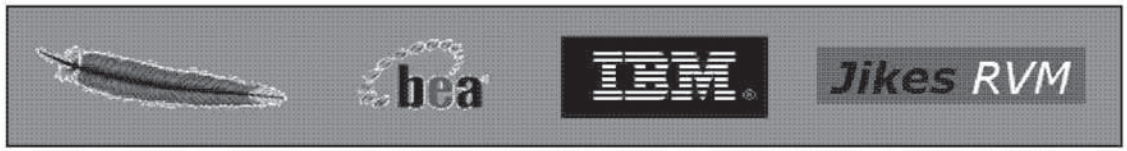
Agenda

- Project History
- Development Model
- Code and Runtime Modularity
- How Are We Doing?
- Demo
- Harmony and the JCK
- Wrap-up

Architectural overview

Everything is pluggable





Harmony DRLVM – the Java Virtual Machine

- Core interpreter
  - Bytecode verifier, class file loading / unloading, debug interface, JVMTI etc.

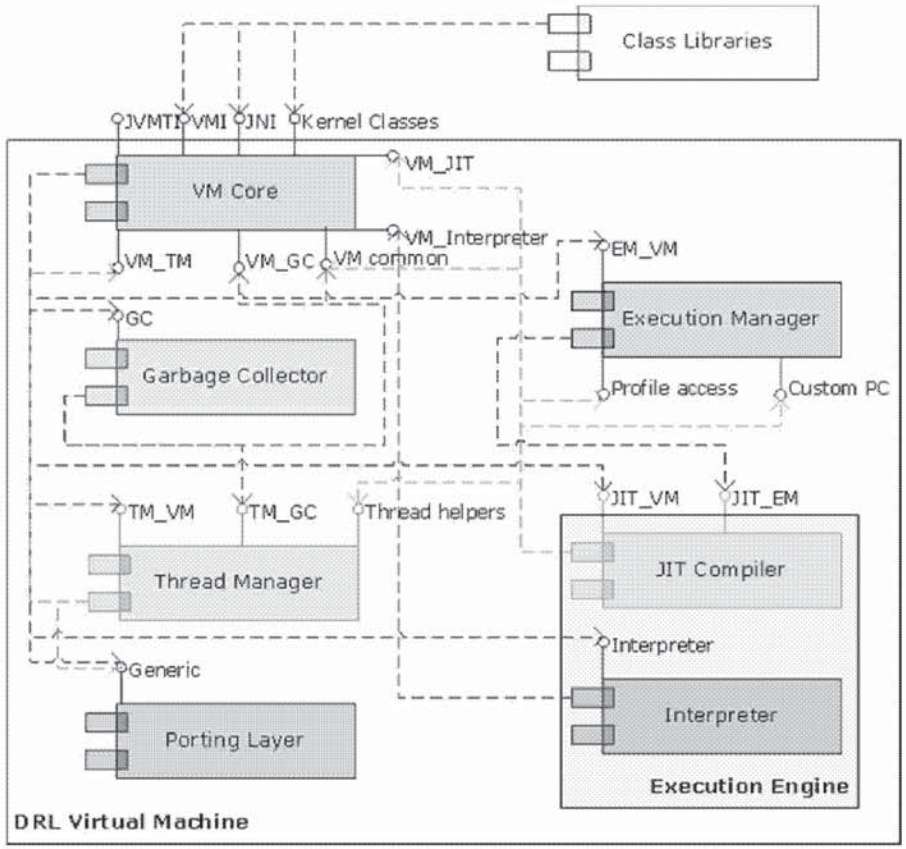
- Just in Time compilers
  - profiling engine selects compiler based on typical code paths, data values, …
  - **"JET" fast compiler**
    - ~14K NSLOC, ~10-20K methods/sec, resulting code 10x faster than interpreter
  - **"OPT" optimizing compiler**
    - multiple code transformations, resulting code 2x faster than JET

- Memory managers
  - "GCv4" stop the world, sequential copying compacting
  - "GCv5" stop the world, parallel copying compacting generational
  - "Tick" on the fly, concurrent mark, parallel, mark-sweep-compact generational

Virtual Machine modularity

Code execution and memory management

- Well-defined interfaces, consistent across platforms
- Interfaces do not compromise runtime performance
- Modules either build-time or runtime replaceable
- Multiple implementations already exist for some modules

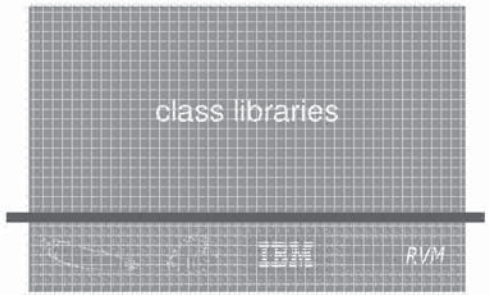


Harmony's VMI

Virtual Machine Interface to the class libraries

- VM-specific 'kernel' classes
  - 23 publicly defined Java SE types that the VM typically knows intimately, plus one helper
- Access to our C struct from JavaVM / JNIEnv pointers
  - 10 new C functions that augment standard JNI

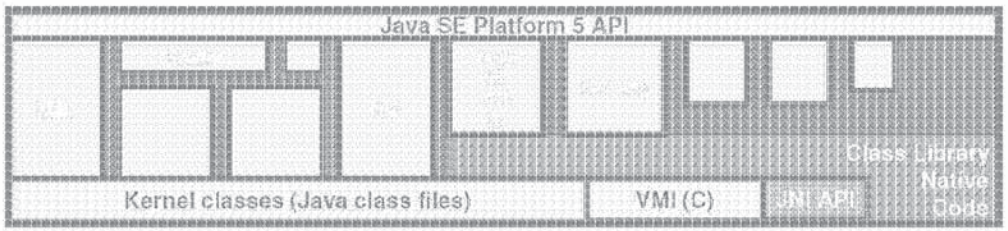
- The VMI provides:
  - Access to the operating system abstraction library (port library)
  - Access to per-VM storage functions (VMLS) which allows multiple VMs to exist in a single address space
  - Ability to get/set/iterate system properties
- The VMI does not :
  - Require any enhanced VM/class library linkage
  - Prescribe object layout, garbage collection, synchronization, and so on



Class library modularization

Java SE platform implemented in ~30 components

- A module…
  - is related functionality scoped by Java packages
  - 'exports' user-API and internal-API, hides private internal implementation
  - defined by dependencies in the Java specification
  - minimizes coupling by explicit internal APIs



  - is delivered as a JAR file, a real OSGi bundle

- Observed benefits
  - easier to manage prior exposure
  - freedom of assembly for module consumers
  - unit of replacement for fixes and updates
  - facilitates contributions

Development time modularity

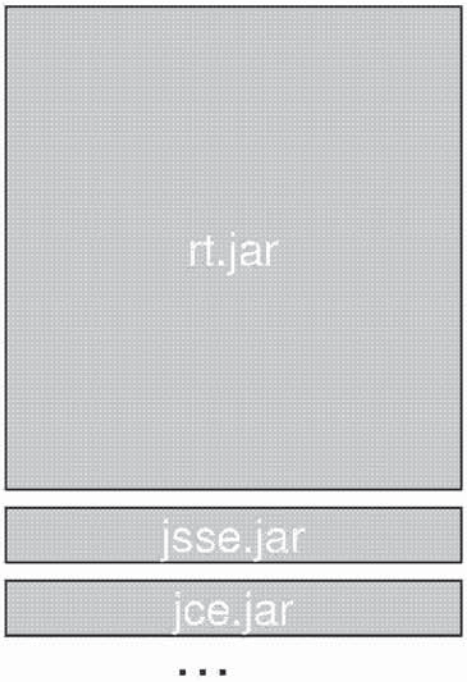
The Harmony Development Kit ("HDK")

- Not a replacement for JDK software
  - HDK → Harmony developers :: JDK software → Java developers
  - Contains all files necessary for Harmony development and testing
- Enables fast rebuild of individual modules – Java platform and native
- Removes necessity to check out whole source tree
- Supports separate or in-place development of HDK trees

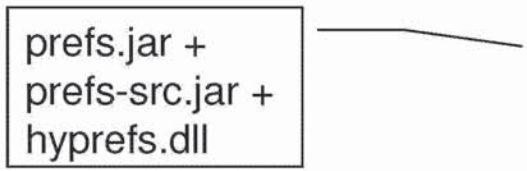
Packaging Modularity

Software assemblies

Conventional approach



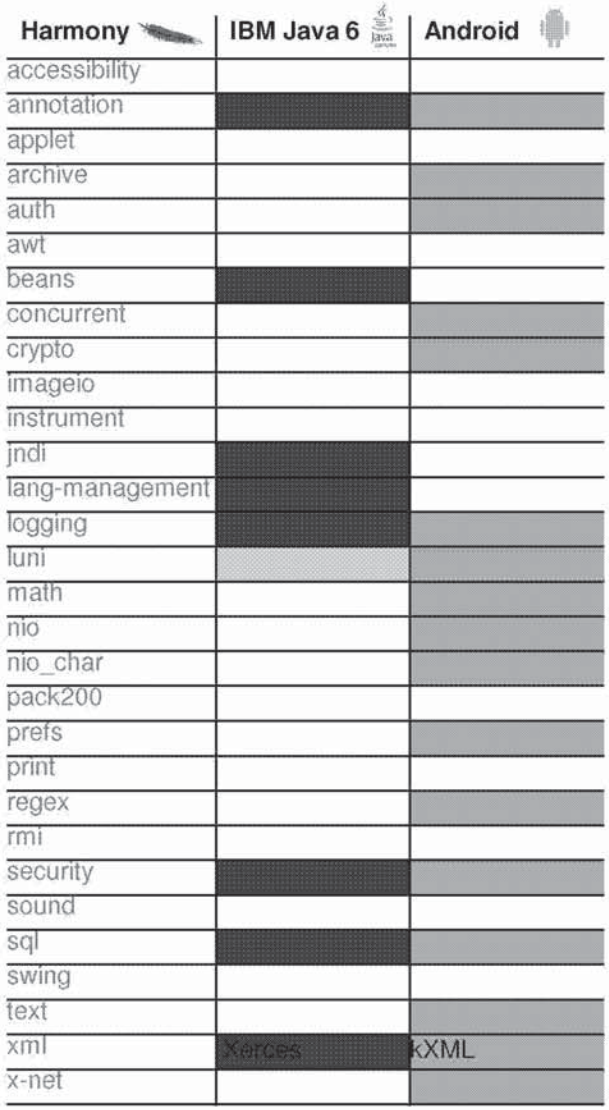
Harmony's approach



Multi-configuration support

"Right sized" application runtimes

- Harmony's modules enable flexible assemblies:
  - Apache Harmony : builds all modules into Java 5 and Java 6 configurations
  - IBM Java 6 SE : a number of modules sourced from Harmony
  - Android : core libraries adapted to run on Dalvik VM



Runtime modularity

Multi-VM and tools launcher

- Single native launcher program
- Runs command-line Java technology programs, including generic launcher and JDK software tools
- Select VM provider based on command-line option
- Select runtime specified modules (e.g. GC algorithm)

```
jre/bin/
   java.exe
   default/
   drlvm-v1/
   drlvm-v2/
   ibm-j9/
$ java MyClass
$ java -vmdir:ibm-j9 MyClass
$ java -vmdir:drlvm-v1 MyClass
```

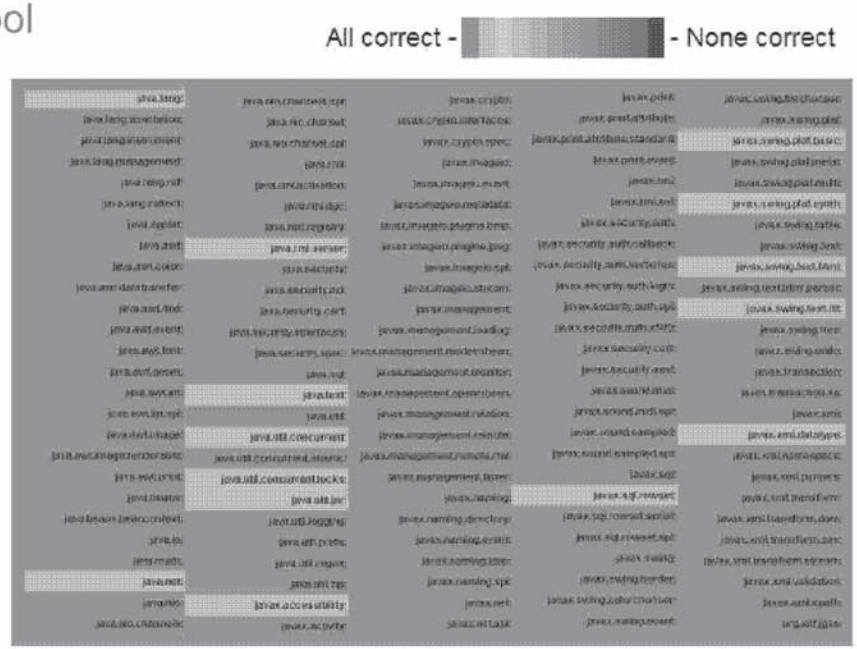
Agenda

- Project History
- Development Model
- Code and Runtime Modularity
- How Are We Doing?
- Demo
- Harmony and the JCK
- Wrap-up

API completeness

Our original goal - Java SE platform 5

- Completeness "heat map" courtesy of JAPI tool
  - compares API compatibility
- Over 99% 1.5 API complete
- Missing functionality is now primarily in…
  - providers
    - sound formats encoding / decoding
    - Kerberos authentication
  - command-line tools
    - javadoc, javaws, jstack, native2ascii, etc.



Comparison	Good	Minor	Bad	Missing	Abs.add
Harmony 5.0	99%	0.16%	0.01%	0.8%	0%
Harmony 6.0	96.74%	0.17%	0.15%	2.91%	0.01%

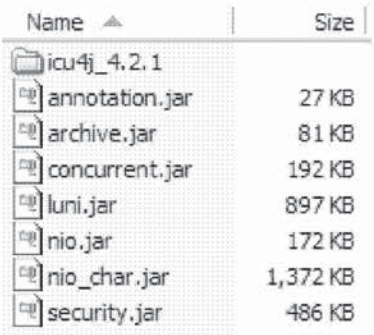
September 2010

See: http://kaffe.org/~stuart/japi

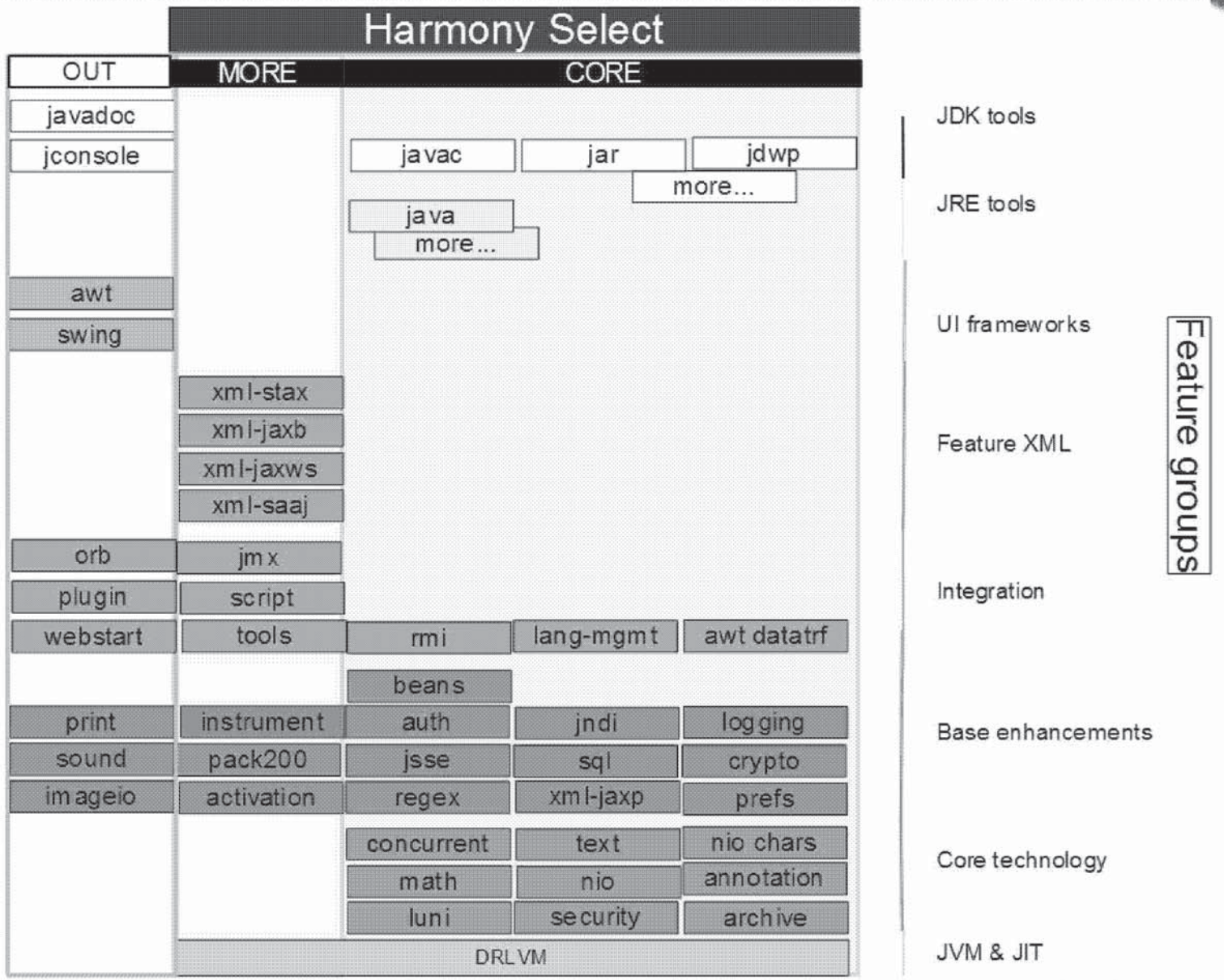
Fit for Purpose Runtimes

- Java technology targeting specific runtime uses
- Mobile devices vs. server-grade services
- Virtualization
  - Cloud computing platform
  - Hypervisor-aware managed runtimes
- Big data
  - Map-reduce optimized runtimes

Trivial headless subset for "Hello World!"



Name	Size
icu4j_4.2.1	
annotation.jar	27 KB
archive.jar	81 KB
concurrent.jar	192 KB
luni.jar	897 KB
nio.jar	172 KB
nio_char.jar	1,372 KB
security.jar	486 KB



Code quality

testing, bug fixing, metrics, ...

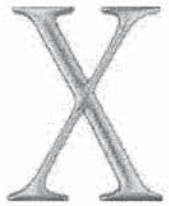
- Harmony suites:
  - Class library unit tests
  - DRL VM unit and regression tests
  - Functional test suite
  - JDKTools unit tests
  - Reliability test suite
  - Stress test suite
  - VM Validation test suite
- Application scenarios:
  - Apache Ant scenario
  - Apache Axis2 client-server scenario
  - Apache Struts scenario
  - Apache Tomcat scenario
  - Dacapo benchmark
  - Eclipse hello world application
  - Eclipse Geronimo application
  - JEdit scenario
  - Jetty scenario
  - LDAP scenario
  - Scimark benchmark
- External test suites:
  - Apache Ant 1.7.0 unit tests
  - Apache Maven unit tests
  - Eclipse unit tests.
  - Eclipse TPTP tests
  - Geronimo unit tests.
  - JUnit tests
  - Mauve test suite

Multi-platform support

Write once, debug everywhere!

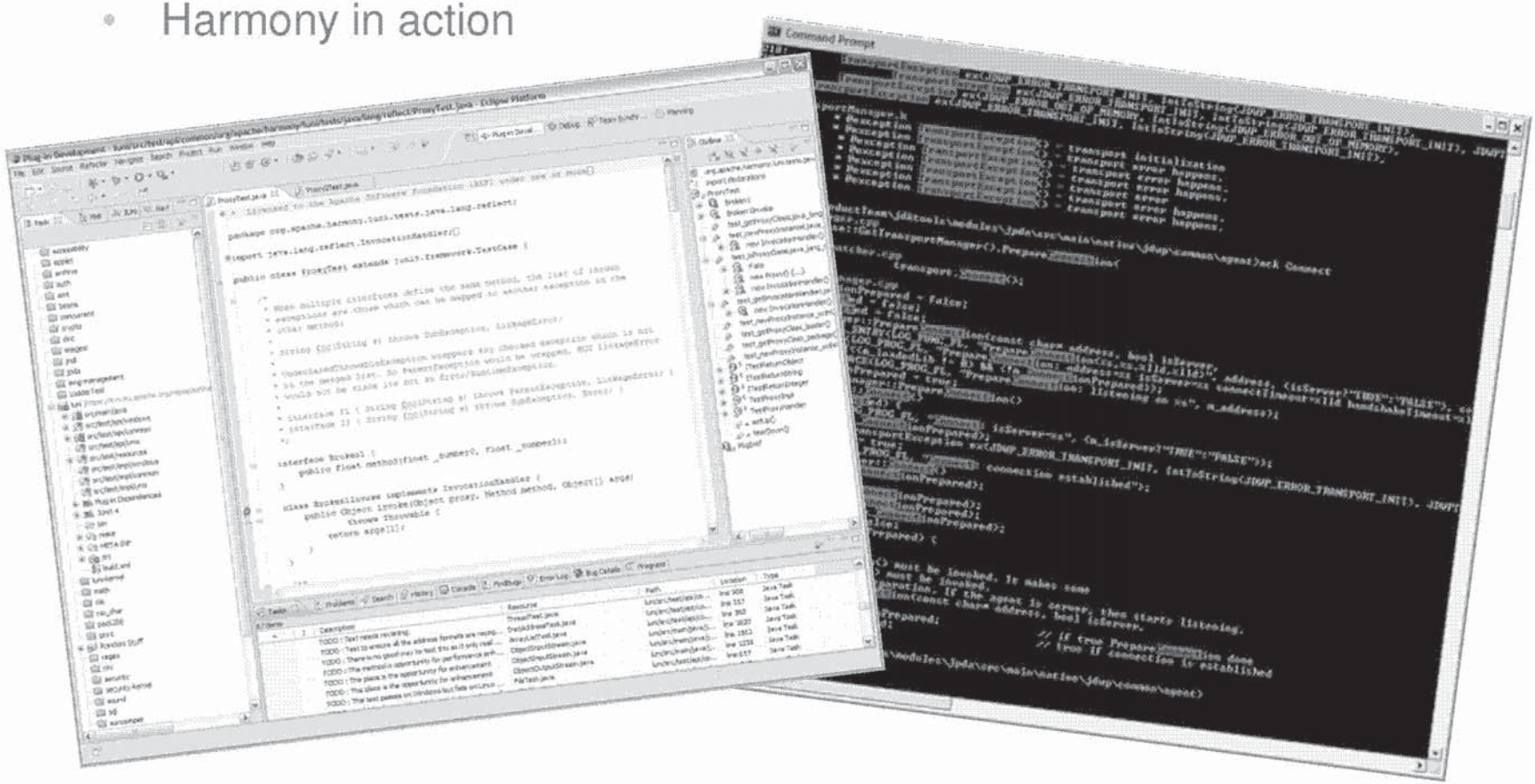
- Community builds, tests, and reports results on :
  - x86 / x86_64 Windows 2000, Windows XP, Windows Server 2008
  - x86 / x86_64 Debian, SLES11, RedHat AL 6, Ubuntu 10
  - PPC 32bit / 64bit RedHat AL6, SLES 11, AIX6.1
  - zSeries 31bit / 64 bit zLinux and zOS V1R12
- Work in progress on :
  - Itanium Linux and Windows
  - x86 Mac OS X and FreeBSD
- Interested in other platforms? Us too!





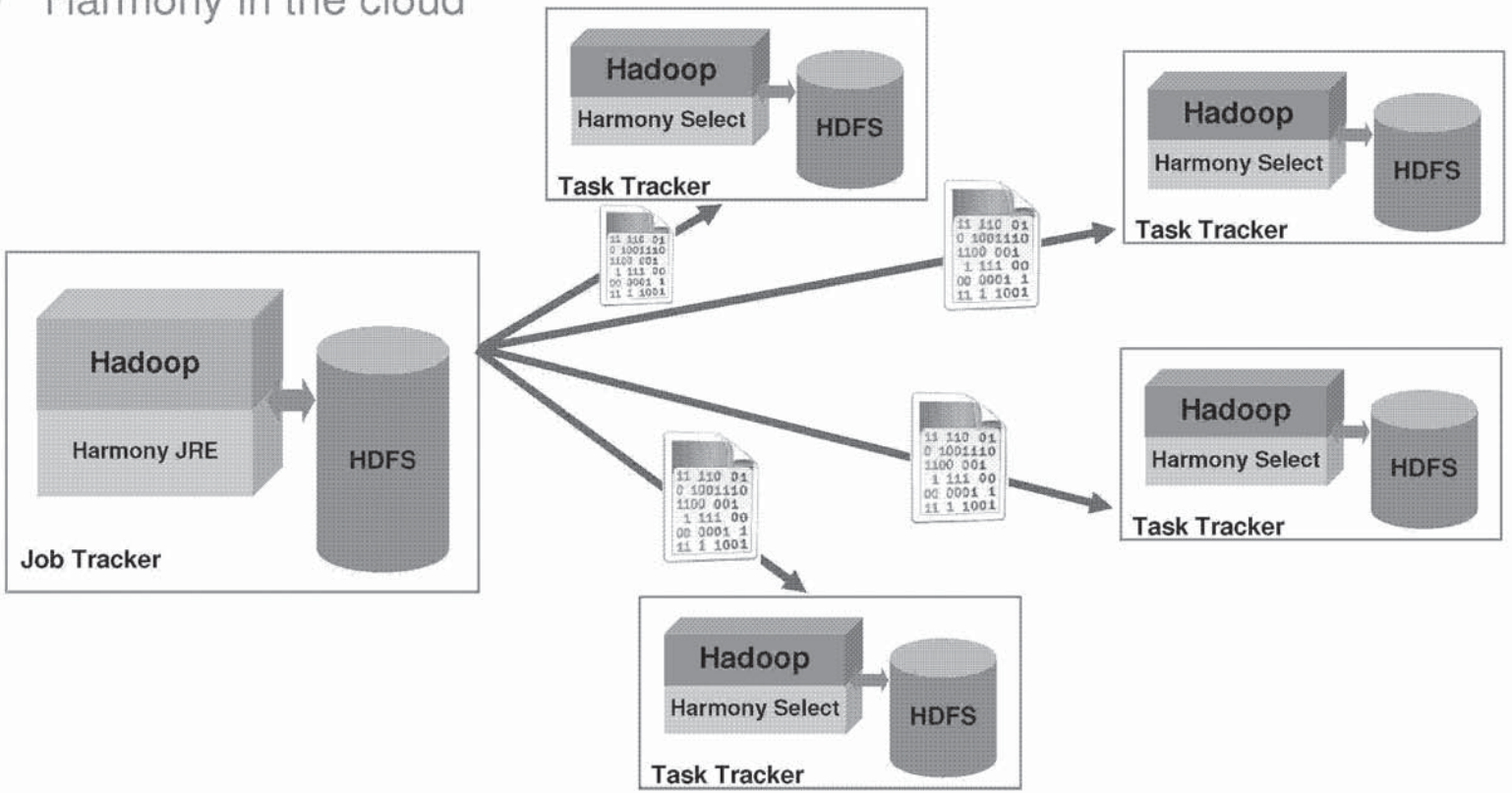
DEMO

- Harmony in action



Apache Harmony + Apache Hadoop

- Distributed compute cluster
  - Nodes running on right sized runtimes
  - Harmony in the cloud



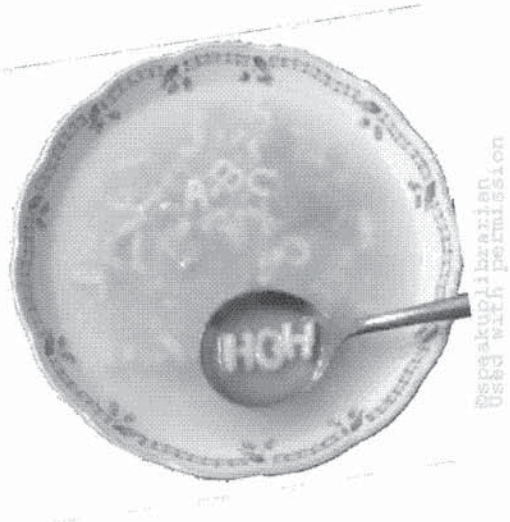
Agenda

- Project History
- Development Model
- Code and Runtime Modularity
- How Are We Doing?
- Demo
- Harmony and the JCK
- Wrap-up

Apache's JCK issue with Sun

History

- First request for Java SE JCK v5 - August 2006
  - As per changes to agreement in JSPA, the "Apache Compromise"
- Sun offered JCK with FoU restrictions – 3Q2006 / 1Q2007
  - Limits how users can use compliant software, ASF reject as not allowing distribution simply under ALv2.
- Open Letter from ASF to Sun – April 2007
  - Appeal directly to Schwartz gets no response.
- ASF position on the JCP – August 2007
  - The ASF will vote "no" on any proposal involving:
  - × JCP lead who is not complying with the JSPA
  - × JSRs where the TCK-license contains FoU restrictions



Apache's JCK issue

ORACLE

- Current position
  - Last Sun offer with new terms for the JCK are still unacceptable.
  - Contains FoU limitations on ASF's usage of the JCK (rather than downstream user restrictions).
    - This would allow us to redistribute under ALv2.
  - However, terms contain the requirement for an 'official' notice from Sun regarding IP notice and certification requirements – effectively a *de facto* augmentation of the Apache License, and therefore unacceptable.

Apache does not make statements regarding intellectual property other than facts we explicitly know to be true and are thereby disclosed in Notice files and the Apache License.

- Oracle not yet offered alternative terms

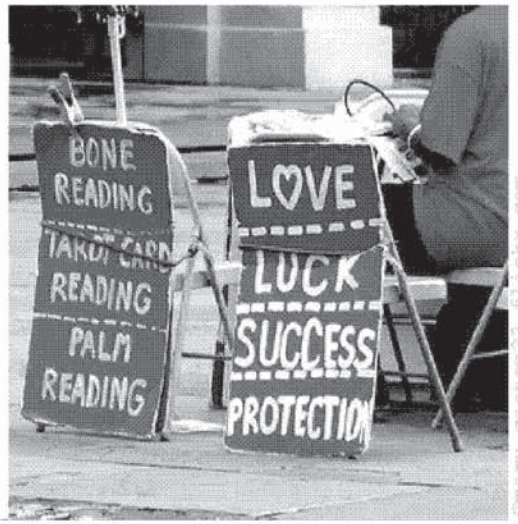
Agenda

- Project History
- Development Model
- Code and Runtime Modularity
- How Are We Doing?
- Demo
- Harmony and the JCK
- Wrap-up

Summary

The best is yet to come...

- We invested time up-front getting the IP infrastructure right.
- Rapidly approaching a full compatible, open-source, implementation of Java SE.
- The core code has been picked up and used in non-trivial commercial runtimes.
- Our modularity story has given us flexibility and robustness to progress quickly and maintain stability.
- Harmony is a fountain of innovation in Java technology.

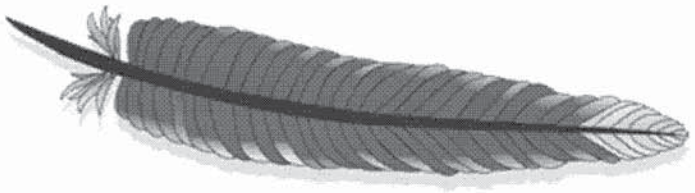


Active areas of effort

What we still need to do … help appreciated

- Class library providers completion
- VM performance work
- Porting to new operating systems
- Improved test case coverage
- Real-world application testing
- Community growth

Apache Harmony



Q&A

Tim Ellison

tellison@apache.org

http://harmony.apache.org